Investigating Agent Behavior In different Reinforcement Learning methods by optimizing strategy to gain as much reward as possible.

Al-Hitawi Mohammed
University Of Fallujah
AI specialist at ELTE

Bertold Pal
Student at BME

Saif Ali
MSc student at BME

Abstract

This report presents methods for pruning different Reinforcement Learning algorithms that show different agent behavior in different models, to facilitate research into understanding the behavior of these strategies. Implementing and vitalizing Dynamic Programming, Monte Carlo (MC), and Temporal difference (TD) algorithms and comparing their results. The pruning algorithm takes a given dataset that has 2 cases to show the shortest path depending on how the agent learns from their behavior where probability are given.

Keywords*: Mechanistic interpretability, ML safety ,Deep Reinforcement Learning , Agents systems , Artificial Intelligence (AI) in Processes and Automation*

1. Introduction

In order to achieve the desired behavior of an agent that learns from its mistakes and improves its performance, we need to get more familiar with the concept of Reinforcement Learning (RL). Implementing such a self-learning system is easier than we may think we already know the agents systems but what is them behavior look like! We will show during an example with experimenting the three algorithms mentioned above. Let us assume we are in Baghdad city and we need to go to Fallujah as fast as we can. There are two roads where we can leave the city (Baghdad), route 10 and 11. After we arrive to Fallujah east, we have only two bridges we can choose from to cross the Euphrates River to arrive to Fallujah west. The traffic is unpredictable, so it can happen that the road or bridge, we choose has a traffic jam. The time we need to cross the bridges depends on the first action (route 10 or 11), so they will be different either case. In addition, sometimes we are redirected even though we have chosen the other route/bridge. Our goal is to build a strategy, where we gain the most reward on our journey. Reward are the negative time we need to go through that road/bridge.

The research question here: Why do these algorithms all perform different? (Algorithms model based vis non-model based)

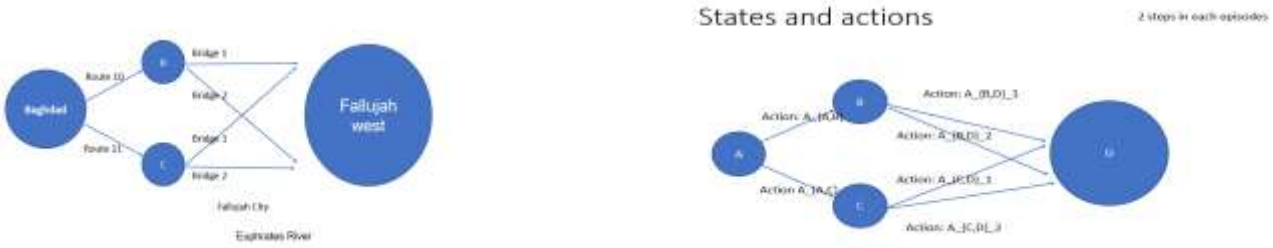


Figure 0 : illustration for our an example

2. Methods

Algorithms model and non-model based. More details in this notebook (https://github.com/Mohammed20201991/Blue_team_MechanisticInterpretability)

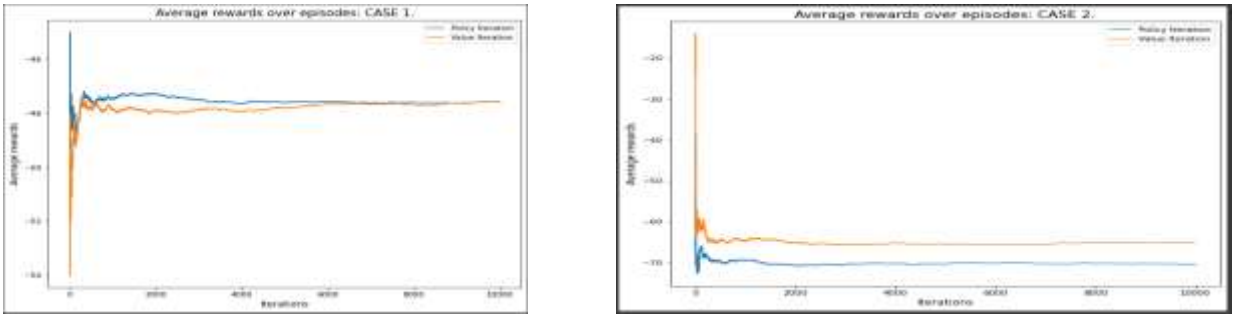
2.1: Policy & Value Iteration

2.2: Monte Carlo

2.3: Temporal Difference

3. Results

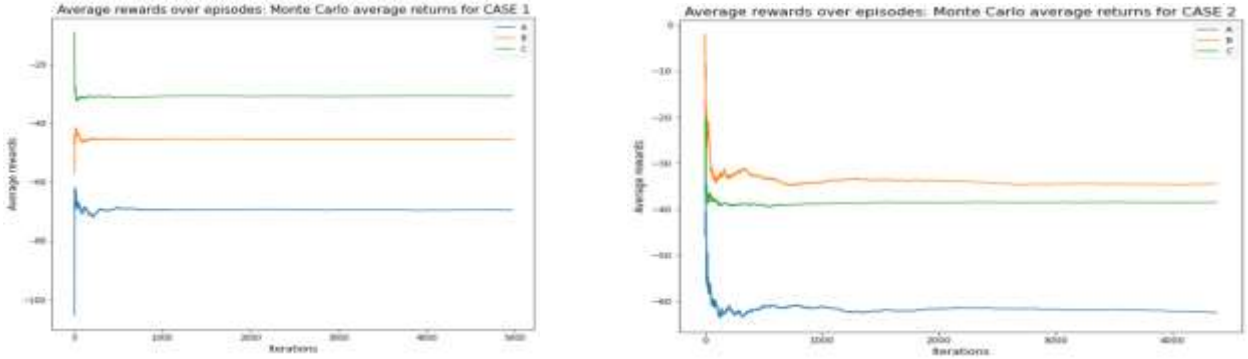
Here we show some cases and results.



Figuer1: Average Reward over Episodes compassion for value and policy iterations in two cases

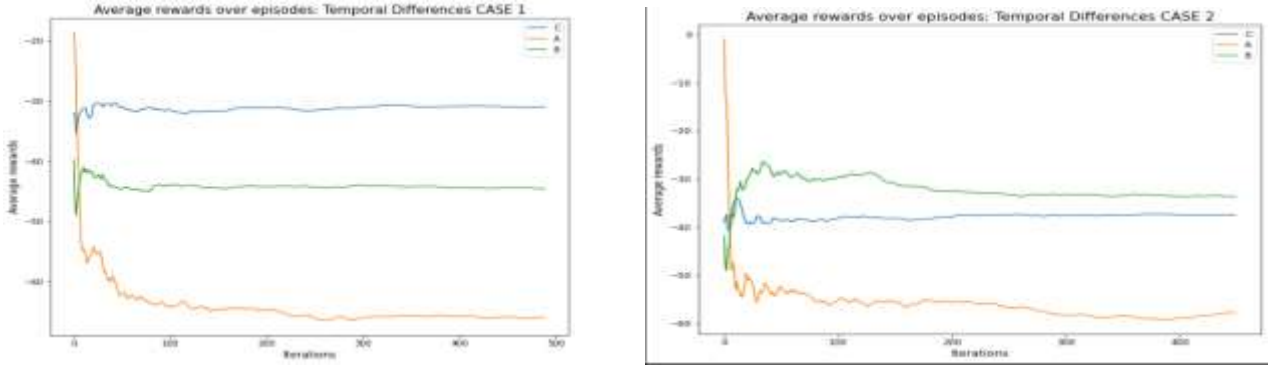
Case : Value and Policy iterations

Case : Monte Carlo



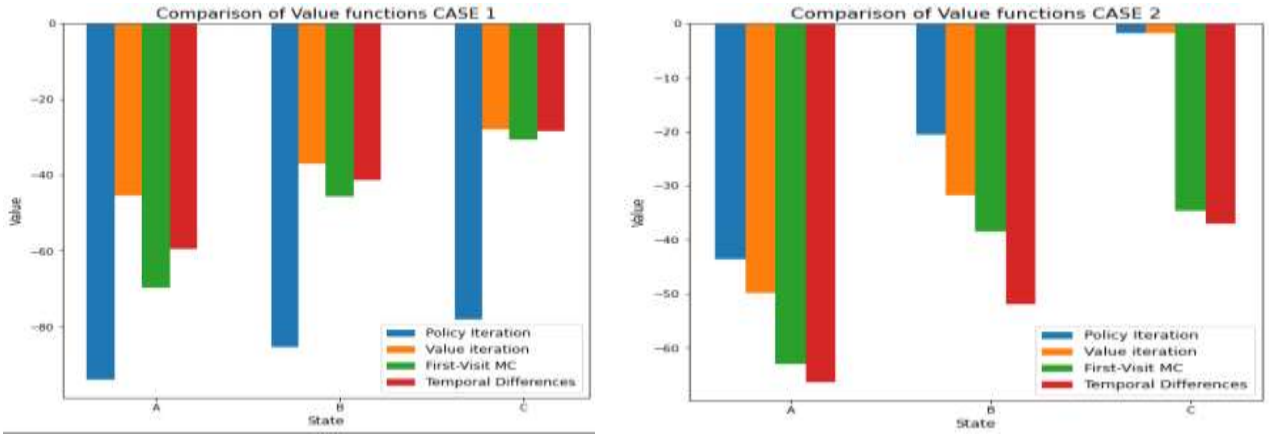
Figuer2: Average Reward over Episodes compassion for Monte Carlo method.

Case : Temporal Difference Methods .



Figuer3: Average Reward over Episodes compassion for Temporal Difference Methods.

More vitulazation :



Figuer4: Comparison value functions.

Figure 1 – Representation of Benchmarking Number Comprehension Conflation

4. Discussion and Conclusion

We have previously seen value functions comparisons and a deep analysis of how the average rewards converge while running policy iteration, value iteration, Monte Carlo and temporal difference learning. They all have different performances because they all do the

searching of the backup in a different way. Policy and Value iteration both start from a given state and from there check all possible states, and evaluate how good they are given that the agent is in a given state. This yielded better results in most of our experience, because how these methods utilize the dynamics of the environment to calculate the values of state-value functions. Monte Carlo estimation is fast and powerful because it does not need to know the probabilities of states and actions. It can learn from pure experience by sampling complete episodes. By this MC is doing a depth search. Temporal Difference learning samples a single step at a time and builds the model by sampling that single step. It is also not necessary to know the transition and action probabilities. The performance of MC and TD were very similar in most cases. Regarding the comparison of the value functions, we can note that policy and value iterations were close to each other in value in almost all the cases. The MC and TD algorithms were also performing in a similar fashion. In the second case, where there was a specific chance of de-routing the agent to a different state, yielded less total reward than in the first case. This uncertainty made the non-model-based algorithm perform worse than the model-based ones. In the first case, the learning always exited with a larger average reward, because there was less unsureness in the rewarding system. This being said, we can surely note that all the algorithms converged, and the learning yielded a sure solution in all the cases. This is concluded as a success.

5. References

- SUTTON, Richard S.; BARTO, Andrew G. *Reinforcement learning: An introduction* . MIT press, 2018.https://alignmentjam.com/post/quickstart-guide-for-mechanistic-interpretabilityhttps://www.youtube.com/watch?v=RlugupBiC6w&ab_channel=CodingPerspective
- https://www.youtube.com/watch?v=4KGC_3GWuPY&t=308s&ab_channel=Zenva
- https://www.youtube.com/watch?v=uiPhlFrwcw8&t=337s&ab_channel=HenryAILabs